Simulation and Implementation of an Open Architecture Controller

Fred Proctor, Will Shackleford, and Charles Yang

National Institute of Standards and Technology
Gaithersburg, MD 20899

Tony Barbera, M.L. Fitzgerald, Nat Frampton,
Keith Bradford, Dwight Koogle, and Mark Bankard

Advanced Technology and Research Corporation
Burtonsville, MD 20866

ABSTRACT

The National Institute of Standards and Technology (NIST) has developed a modular definition of components for machine control, and a specification to their interfaces, with broad application to robots, machine tools, and coordinate measuring machines. These components include individual axis control, coordinated trajectory generation, discrete input/output, language interpretation, and task planning and execution. The intent of the specification is to support interoperability of components provided by independent vendors. NIST has installed a machine tool controller based on these interfaces on a 4-axis horizontal machining center at the Pontiac Powertrain Division of General Motors (GM). The intent of this system is to validate that the interfaces are comprehensive enough to serve a demanding application, and to demonstrate several key concepts of open architecture controllers: component interoperability, controller scalability, and function extension. In particular, the GM-NIST Enhanced Machine Controller (EMC) demonstrates interoperability of motion control hardware, scalability across computing platforms, and extensibility via user-defined graphical user interfaces. An important benefit of platform scalability is the ease with which the developers could test the controller in simulation before site installation. The EMC specifications are serving a larger goal of driving the development of true industry standards that will ultimately benefit users of machine tools, robots, and coordinate measuring machines. To this end, a consortium has been established and cooperative participation with the Department of Energy Technologies Enabling Agile Manufacturing (TEAM) program and the United States Air Force Title III program has been undertaken.

1. INTRODUCTION

In the early 1990s, the Manufacturing Engineering Laboratory of the National Institute of Standards and Technology (NIST) began the Enhanced Machine Controller (EMC) program to develop a modular definition of components for machine control [1]. The intent was to document the interfaces to these modules to the degree that would allow independent third parties to provide interoperable products. The development of this modular architecture grew out of NIST's involvement with the United States Air Force's Next Generation Controller (NGC) program, and the Real-time Control System (RCS) architecture [2, 3] which has evolved within NIST over many years. In a unification of RCS with the NGC architecture as embodied in the Specification for Open System Architecture Standards (SOSAS) [4], a definition of the modules which make up a controller and their interfaces was developed. This architecture is shown in Figure 1.

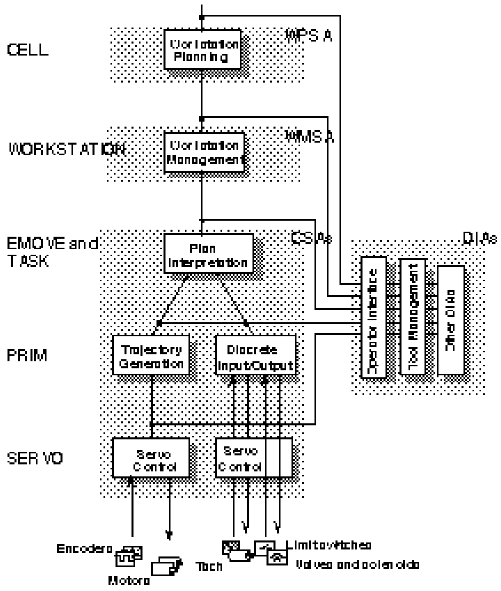

Figure 1. The Enhanced Machine Controller Architecture. The labels on the left, CELL through SERVO, indicate nomenclature from the NIST RCS reference model. WPSA, WMSA, CSA, and DIA represent modules from the NGC SOSAS. The EMC architecture is derived from an RCS analysis of machine controllers, and is consistent with the SOSAS model of standard applications.

In this figure, boxes indicate the individual modules for which interfaces have been defined: workstation planning, workstation management, plan interpretation, trajectory generation, servo control, and discrete input/output. These have been mapped onto the broader NGC SOSAS "Standard Applications" (SAs), e.g., Workstation Planning (WPSA), Workstation Management (WMSA), and Controls (CSA). The EMC architecture defines modules at a higher granularity within the CSA. This reflects both the existence of products at this granularity, as well as their presence in the NIST reference model architecture. The modules on the side indicated by DIA are Domain Independent Applications, in SOSAS parlance. These modules do not require any specific interfaces themselves, but may be developed using only the interfaces provided by the other modules. For example, a necessary module for a controller is the operator interface. However, implementations of the operator interface need only avail themselves of messages to the controller and data provided by the controller: no additional interfaces are required to be defined in order to incorporate an operator interface into an EMC controller.

The interface specifications are formalized in the C++ programming language, using header files. The specification consists of messages into each module, and world model data provided by each module. Both the messages and world model data are implemented using C++ classes. Underlying the EMC architecture is a model of data transfer, the Neutral Manufacturing Language [5] (NML). This model provides for "mailboxes" of data, with one or more readers and writers. The control methodology indicates that control commands into each module originate with a single writer, and are read by a single reader. World model data originates with a single writer, but may be read by arbitrarily many readers. For example, both the trajectory generator module and the operator interface may read the current position of the actuators as maintained by the servo control module.

The EMC specifications have been implemented on a variety of machines, both at NIST and elsewhere. In this paper, we will describe an implementation built for the General Motors (GM) Powertrain facility in Pontiac, Michigan, retrofitted to a Kearney and Trecker (K&T) 800-series horizontal machining center. The intent of this installation was to provide a validation testbed for the interfaces. In our laboratory implementations of the EMC, we often found ourselves deferring the solution to problems that were not germane to the research goals. In the case of the definition of EMC interfaces, however, we found that our laboratory systems only took us so far, and we needed a true production application to gain confidence that the interfaces would be sufficient and correct.

Due to the geographic separation between NIST in Maryland and GM in Michigan, we found it necessary to build a high-fidelity simulation of the K&T machining center so that we could develop most of the controller locally. Additionally, the requirement that the machine be available for use on a weekly basis by the GM machinists meant that we needed our own development simulation during the machine's use. Installation of a series of connectors allowed us to quickly swap between the existing controller and the EMC, so that the development cycle consisted of local coding and testing on the simulator, controller swapover and test, and swapback. This provided both the development team and the machinists reasonable access to the machine during the entire development phase of the GM EMC.

It is important to note that no portion of the controller was simulated: only the machine to which the controller was installed. Simulating any portion of the controller would introduce uncertainties when that portion was run on the production machinery. Commercially available software applications for simulating the machining process or motion control are widely available, but were of little use during controller development since these products assume ideal controller behavior and are used to validate the controller input. The exact opposite situation applied to the EMC controller development: we assume that the controller inputs (i.e., numerical control programs) are correct, and the controller itself needs to be proven.

2. SYSTEM DESCRIPTION

Before detailing the simulation of the EMC, it is worthwhile to describe the actual machining center itself, particularly the physical interfaces to which the EMC installs. The Kearney and Trecker 800 horizontal machining center has 4 axes of coordinated positioning. These axes are typically labeled X, Y, Z, and B. The X axis is horizontal, and moves side to side when looking at the front of the machine into the spindle tooling. The Y axis is vertical, and moves up and down. The Z axis moves in and out, parallel to the axis of spindle rotation. The B axis is a rotary axis, whose axis of rotation is parallel to the up-and-down Y axis. Physically, it sits atop the X axis, and is level with the floor similar to an audio turntable. The X and B axis form one kinematic chain, and the Z and Y axes form a second. That is, the workpieces are affixed to the B axis, which moves with the X; the spindle is affixed to the Y axis, which moves with the Z.

The spindle is position- or velocity controlled, providing up to 3000 revolutions per minute (rpms) of speed through 3 gear ranges. During cutting, velocity is maintained using tachometer feedback. During positioning, for example to orient the spindle during a tool change, resolver feedback is used to precisely orient the spindle axis. In total, there are 5 axes of servo control: 3 linear axes, 1 rotary, and the spindle.

The K&T 800 also has a tool change mechanism for 68 tools and a pallet shuttle which allows pallets of workpieces to be loaded on or off the machine. These are controlled by sequencing outputs to solenoids and hydraulic valves in response to sensed limit switch closures. Altogether, there are 38 discrete outputs and 66 discrete inputs.

In order to maintain the uptime of the machine, we required a quick changeover procedure that would allow us to convert to our development controller upon our arrival, and to switch back to the original controller when we left. The staff of General Motors provided engineers who designed the interfaces necessary for this quick change procedure. Briefly, the procedure consisted of removing the original resolver feedback from the X, Y, Z, and spindle axes and connecting them to the resolver-to-encoder converter inputs of the EMC; removing the amplifier outputs to the X, Y, Z, and spindle axes and connecting them to the EMC outputs; and removing the computer logic ribbon cables for the discrete input/output points and connecting them to the EMC digital input/output boards. The B axis was left alone during the changeover, due to the incompatibility of its electronics. Instead, we accessed its position feedback and velocity output on the same digital interface that contained the discrete input/output information. This necessitated a software servo, as described in a subsequent section.

In order to adequately test the controller locally, all of these components required simulation. Whenever possible, the simulation was performed with hardware as similar to the real hardware as possible. Ideally, we would have liked to have an entire identical machine installed in our laboratory. In practice, this was too expensive and would have taken far too much installation time. In other cases, we simulated portions of the machine in software, taking great care to characterize the system's timing during initial visits to the production facility. Of course, much of the initial controller development followed the time-honored practice of converting physical output signals to printed output. While suitable for initial tests, this practice was almost useless during most of the development since many of the problems encountered in real-time systems originate with timing.

The EMC simulation system consists of 4 components: a small 3-axis milling machine which physically simulates the machining center and the signal inputs and outputs to the axes; a duplication of the operator interface, including pushbuttons and lights; a personal computer simulation of the discrete input/output points of the controller which provides an identical interface to the input/output points on the real machining center; and a graphical animation of the machine. These components, shown in Figure 2, are detailed in the following sections.

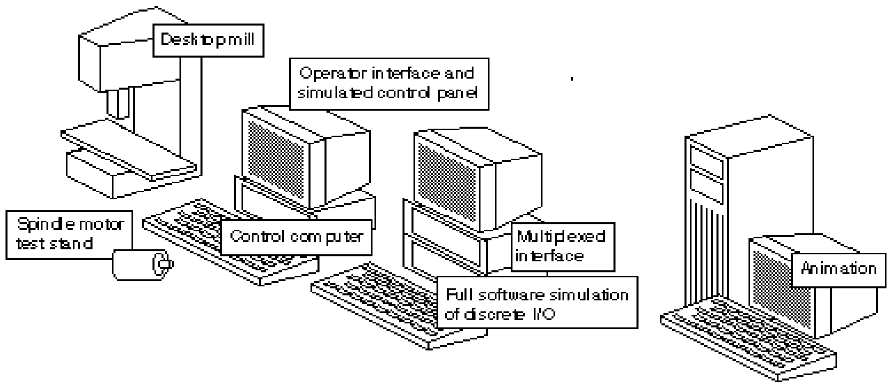
Desktop mill
Operator interface and simulated control panel
Spindle motor test stand
Control computer
Multiplexed interface
Full software simulation of discrete I/O
Animation

Figure 2. Simulation setup of the EMC, with cabling omitted for simplicity. The output of the control computer is split between the desktop mill/spindle motor and the multiplexed interface to the software simulation of the discrete input/output. Parts can be machined on the desktop mill and verified visually. Tool changing and pallet shuttling are verified via the animation, which displays the state of the software I/O simulation in real time.

3. AXIS SIMULATION

The machine's linear axes were simulated with a desktop milling machine originally intended for prototype or model making. As sold, the unit provided stepper motor control which had to be replaced with servo motors and amplifiers to more closely duplicate the K&T. Further complicating the control was the fact that the original axis feedback devices on the K&T were resolvers. Because the desktop milling machine provided encoder feedback, and the motion control boards we selected for the EMC required encoder feedback, we decided to install resolver-to-encoder converters on the K&T itself. Once this was done, our interface to the axis servo control of the desktop machine was identical to that of the production machine.

However, some important differences remained. The minimill was a vertical machining center while the K&T was horizontal; the kinematic configuration of the Y axes differed; we had no B axis on the simulated mill; and the homing and overtravel limit switches on the K&T were not available to the EMC motion control boards. The horizontal- and vertical differences were of no consequence for the control. More importantly, on the desktop mill, the X and Y axes lay on the same kinematic chain which contained the workpiece fixture, while the Z axis supported the spindle. On the K&T, the Y axis was on the spindle's kinematic chain along with the Z. Simulating the same kinematic configuration would have meant significant rework on the desktop mill's support structure, or purchasing a different mill altogether. In this case, we decided that the uncertainty introduced by this difference would have little impact, since the kinematic parameters were localized in the controller software and were easily modifiable. In fact, the difference in this case allowed us to verify that our attempt to localize kinematic parameters such as these actually worked.

We elected not to simulate the B axis since we estimated that the time it would take to measure the dynamics of the B axis and simulate it would exceed the time we would need to tune the control parameters on-site. The situation with the homing and limit switches was entirely different. In this case, not having this feedback on the K&T meant that we would suffer significant lag time between when these switches tripped and when our separate I/O controller could report the trippings to the motion control boards. Therefore, we decided to modify the wiring on the K&T so that these inputs were available directly to the EMC motion control system. Of course, this rewiring was done so as not to impact the original controller. This solution simplified our simulation, since now both the desktop mill and the production machine had identical interfaces to the X, Y, and Z axes.

4. SPINDLE SIMULATION

Like the B axis, the spindle in our desktop mill differed from that on the production machine. First, the spindle on the production machine had tachometer feedback, which our desktop machine lacked. Second, the spindle operated in 3 gear ranges, which we lacked on the desktop machine. Both of these disparities were rectified on the testbed at NIST.

To solve the spindle tachometer problem, we disconnected the controller's spindle interface from the desktop mill, and installed it onto a separate motor test stand consisting of a motor, amplifier, encoder, and tachometer. This provided the identical interface to the real spindle. However, this rendered the actual cutting head on the desktop mill uncontrolled. During cutting tests, the spindle test stand had to be viewed as the cutting spindle, which proved difficult. We also were required to turn the cutting spindle on and off from a separate manual source. In defense of this setup, the system proved quite adequate when developing the interface to controlling spindle speed and orientation. Criticizing this setup, it proved difficult to verify that tapping operations (which require synchronized Z axis motion and spindle rotation) were correct. Like the tuning of the B axis parameters, we left development of this feature of the controller to the on-site integration.

Simulating the interaction of the spindle control with gear changing followed naturally from our high-fidelity simulation of the discrete I/O system, which is detailed in the following section. Anticipating this simulation, we were provided simulated sensed values for the gear engagement, whose timing was precisely measured and simulated. Verifying that spindle gearing changes occurred properly was simply a matter of individually verifying that the controller could access the state of the gear engagement, and visually ascertaining that the required spindle speed sequences occurred at the proper time during the gear shifting.

5. DISCRETE INPUT/OUTPUT SIMULATION

Due to the number of discrete inputs and outputs, and the difficulty in replicating the custom mechanical structures for the tool changer and pallet shuttle, a full software simulation of the discrete input/output system was undertaken. The intent was to provide to the controller developers the identical hardware interface that was present at the machine site. Recalling the controller changeover procedure, this interface was a set of ribbon cables on which were provided all the discrete input/output points that would be connected to a commercial PC-bus card in the EMC. In order to remove any uncertainty about this interface, a full hardware interface including the multiplexing hardware was built. On the simulation side of this hardware interface was a standalone PC running a full software simulation of the discrete I/O system, containing the time delays and dependencies between the limit switches and valves as characterized by timing studies performed early in the installation. This system is depicted in Figure 3.

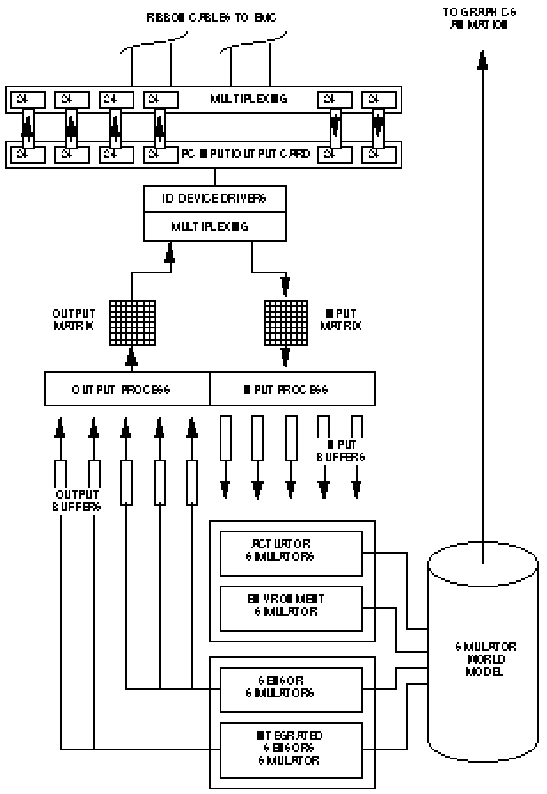

Figure 3. Discrete I/O Hardware Simulator. System consists of a duplication of the multiplexing interface to the actual machine tool, a software simulation of the timing for the discrete actuators and sensors, the simulated world model of the discrete I/O (e.g., tool changer, spindle gearing, pallet shuttle) and a serial connection to the real-time animation running on a graphics workstation.

This system provided immense aid in developing the programs for control of the tool changer and pallet shuttle mechanisms. This was particularly evident when comparing the development cycle with that in previous projects. For example, assuming the programmers had access to a complete machine tool, programming errors may have caused mechanism crashes and physical damage. On the simulator, their effects were innocuous. Furthermore, in cases of complex mechanism interaction, it is easy to put the mechanisms into a state requiring a tedious step-by-step reversal of the failed sequence. Restarting the test after a failure meant at most a simulator reboot.

Of course, software simulation of the entire I/O system reduced the confidence that the controller will perform on the real machine as it does on the simulation. In this case, however, the laborious and costly task of duplicating the physical tool changer and pallet shuttle (perhaps only possible by purchasing a full machine tool) indicated the use of such a simulation.

6. OPERATOR INTERFACE

The interface used by the machinists to run the machining center consisted of a PC-compatible monitor with touch screen, an industrial keyboard and mouse, and a set of lighted pushbuttons and rotary switches. With the exception of the pushbuttons and switches, all these components were easily duplicated with off-the-shelf components for the local simulator. A touch screen was omitted, since the mouse provided redundant capabilities.

The panel containing the pushbuttons and switches required the fabrication of a driver chassis, which converted the 24 Volt signals to 5 Volt signals compatible with computer boards. This driver chassis remained at the local development site until final installation, allowing the programmers to develop the code to read the switches and light the lights locally. When the controller was ready for installation, the driver chassis and associated front panel were shipped to the installation site. This meant that for any further local debug or development, the programmers were without a way to operate the controller.

This unacceptable situation led to the immediate development of a graphical surrogate to the control panel, which was accessible via one of the screens on the graphics display. This simulated control panel provided all the inputs and outputs of the hardware control panel, while saving the cost and time which would have resulted had a new driver chassis and front panel been built. However, the simulated control panel provided no support for the development of the code which interfaced to the hardware driver chassis. For example, we could not test the addition of a new hardware switch or modifications in the lighting patterns. Fortunately, there have been no changes to the control panel design, and this shortcoming of the simulation system has not been an impediment.

One positive side effect of the development of the graphical control panel was recently discovered: one of the transformers in the driver chassis failed, and the machinist continued operation of the machine using the redundant graphical version while technicians repaired the driver chassis.

7. ANIMATION

Supplementing the discrete I/O simulation is the graphical animation. Unlike the desktop milling machine and motor test stand for the spindle, the software simulator has no visibly verifiable output. What is desired is an instantaneous snapshot of the state of the I/O system, particularly in cases where the actuator and sensor states can be correlated with the obvious position of mechanical components. Using a library of graphics primitives developed during previous projects, an animation of the K&T machine tool was quickly built. The positions of the tool carousel, tool changer components, and pallet shuttle were continuously updated based on the model of the world maintained in the standalone PC simulation. This provided almost instantaneous visual verification of the control sequencing. Coupled with the hardware simulation of the machine's axes and spindle using the desktop milling machine and spindle test stand, the full simulation system provided an invaluable visual confirmation of the controller code under development.

8. SUMMARY

Based on the experience of developing a complex controller for a machine located at a remote facility, in almost continuous use, the aid of a high-fidelity simulation has proven invaluable. In particular, care was taken to simulate only the machine being controlled, so that the controller under development was almost plug-compatible with both the simulation system and the real machine. With this as the case, the developers maintained a high degree of confidence that the programs written to control the machine and verified on the simulator would run correctly during the scarce time available for testing. Some points evidenced during this development:

* Full hardware simulation (e.g., machine duplication) gives the most confidence but introduces the possibility of machine damage resulting from failed tests;

* Full software simulation speeds the development cycle, but reduces confidence that the controller will perform on the real machine as it does on the simulation;

* Simulation of electronic interfaces (e.g., resolver-to-encoder converters or multiplexers) introduces almost no risk of physical damage, while greatly increasing the likelihood that the controller will operate on the real machine as it does on the simulated version;

* Use of easily obtainable hardware simulators (e.g., desktop milling machines or motor test stands) is advantageous. Building physical simulations of custom hardware (e.g., tool changer) may provide additional confidence, but will increase the risk of damage and add to the cost and development time.

9. REFERENCES

1. Proctor, F. M., and Michaloski, J., "Enhanced Machine Controller Architecture Overview," NIST Internal Report 5331, December 1993.

2. Albus, J. S., "Outline for a Theory of Intelligence," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 21, No. 3, May/June 1991.

3. Albus, J. S., Lumia, R., Fiala, J. C., and Wavering, A. J., "NASREM: The NASA/NBS Standard Reference Model for Telerobot Control System Architecture," *Proceedings of the 20th International Symposium on Industrial Robots*, Tokyo, Japan, October 4-6, 1989.

4. Martin Marietta, "Next Generation Controller (NGC) Specifications for an Open System Architecture Standard (SOSAS)," National Center for Manufacturing Sciences, 1994.

5. Shackleford, W., and Proctor, F. M., "The Neutral Manufacturing Language Communication System," NIST Internal Report (pending publication), 1995.

Published in Modeling, Simulation, and Control Technologies for Manufacturing, Volume 2596, Proceedings of the SPIE, October 1995.